EE219 Large-scale Data Mining

Course Project #4

Regression Analysis

Shijun Lu（905035448）

Wenfei Lu（505035450）

Xingyi Chen（205032924）

Hanren Lin（304944990）

Introduction

Regression analysis is a statistical procedure for predicting and building the model which evaluates the relationship between target variable and potentially relevant variables. In this project, we used regression analysis with several regression techniques to conduct over-fitting, cross-validation for over-fitting's performance test, and regularization, making backup size prediction.

The dataset used in this project is Network Backup Dataset, which is comprised of simulated traffic data on a backup system over a network. One example data point (source: row 301 in dataset) is shown in form 1.

Week #	Day of Week	Backup Start Time	Work-Flow-ID	File Name	Size of Backup (GB)	Backup Time (hour)
1	Tuesday	17	Work_flow_3	File_22	0.001564649176	1

Form 1 – Example Data Point

Dataset Loading

The first step of the project is dataset loading. For obtaining some further details about the relationships between the categories and patterns of data, we plotted 2 figures for a visualized result. First, we plotted the backup sizes versus day number figure for all workflows, with a twenty-day period applied. The plotted results for 20-day period are shown in Figure 1-5.

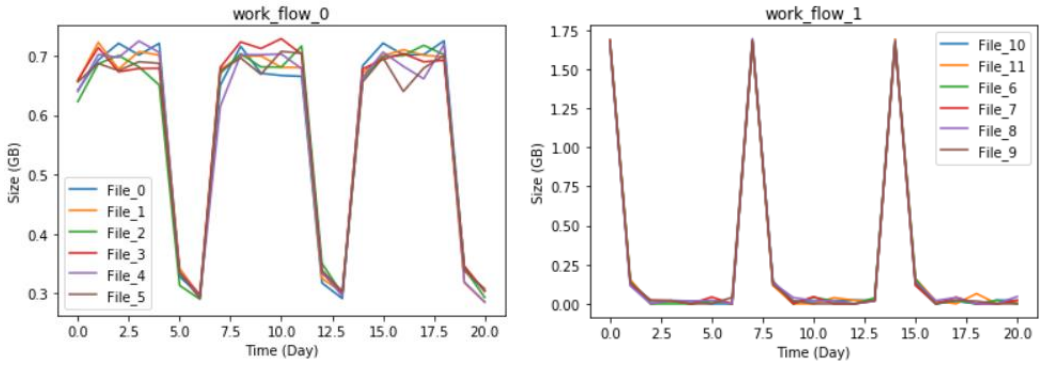

Figure 1, 2 – Plotted result of size versus time (20-day period, workflow 0 and workflow 1)

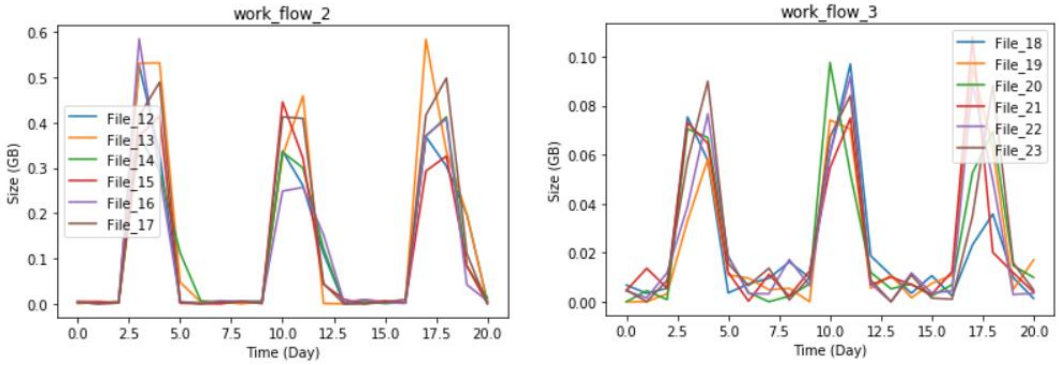

Figure 3, 4 – Plotted result of size versus time (20-day period, workflow 2 and workflow 3)

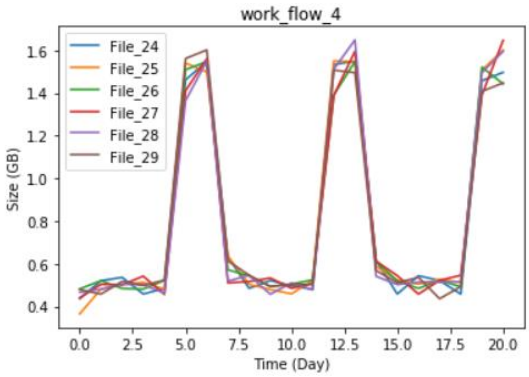

Figure 5 – Plotted result of size versus time (20-day period, workflow 4)

Second, we repeat the above plotting procedure with a 105-day period. The plotted results for 105-day period are shown in Figure 6-10.

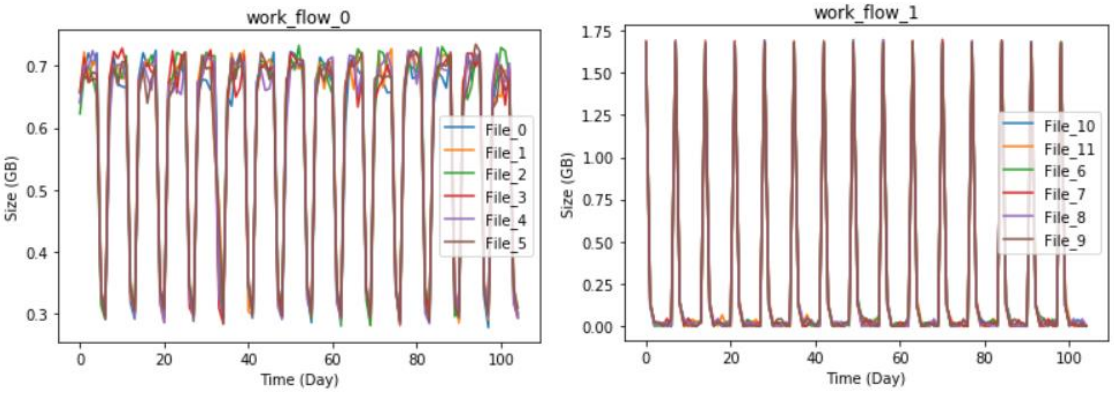

Figure 6, 7 – Plotted result of size versus time (105-day period, workflow 0 and workflow 1)

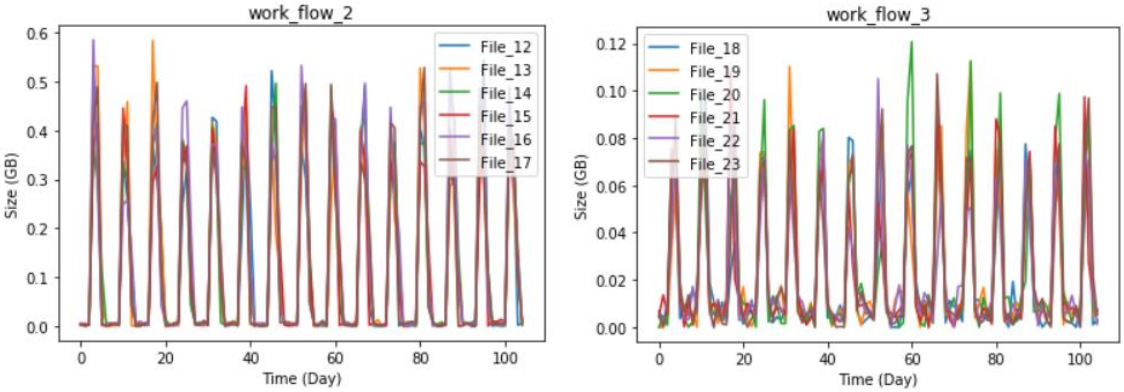

Figure 8, 9 – Plotted result of size versus time (105-day period, workflow 2 and workflow 3)

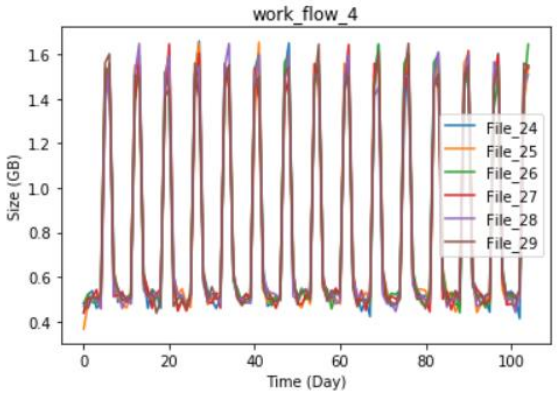

Figure 10 – Plotted result of size versus time (105-day period, workflow 4)

From the plotted results shown above, the figures for both 20-day period and 105-day period have similar repeated patterns. Both figures have several extremums, with extreme points (defined with time) appearing with a certain cycle. In addition, from the 20-day period figures, we could acknowledge that the extremums appear two in a row with a "cycle". For workflow 0, it has the partial minimum values, while all other 4 workflows have the partial maximum values.

Backup Size Prediction

The initial task in this project is to predict the backup size for each data point via other given attributes in various methods, with each method's performance evaluated by reporting training and test RMSE from 10-fold cross validation.

The prediction is based on different algorithms and feature sets. For algorithms, we would apply several regression models, such as linear regression, random forest regression, neural network regression, and k-nearest neighbor regression. For each feature set, it refers to different encoding combination of these 5 features: Day of the week, hour of the day, work-flow number, file-type and week number. The choice of feature set encoding would result in different performance.

The categorical variables could be encoded into one dimensional numerical value. For example, we could convert Day of the Week into one-digit value such as 1, 2, ⋯, 7. This encoding scheme is scalar encoding. In addition, we could also apply One-Hot-Encoding to the categorical variables, which take one of M values, into an M dimensional vector. Since the Day of the week could choose only one value from the set "Monday, Tuesday, ⋯, Sunday", then the corresponding encoded vector can be expressed as [1, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0], ⋯, [0, 0, 0, 0, 0, 0, 1].

For all 5 features, we would apply One-Hot-Encoding to one subset, and scalar encoding to the rest of features, which means each feature could choose one of the two encoding methods. Therefore, the total number of encoding scheme combination is $2^5 = 32$.

Part (a) Linear Regression Model

The first model we applied is linear regression model. Linear regression method models an estimated-linear relationship between a scalar dependent variable y and one or several independent (explanatory) variables denoted x. For project use, we used ordinary least square as the penalty function, which is written as:

$$\min_{\beta} ||Y - X\beta||^2$$

, where the minimization is on the coefficient vector β.

First, we converted each categorical feature into one dimensional numerical values with scalar encoding. The converted features were used to fit the basic linear regression model. For the performance evaluation, we plotted 2 figures, with one having fitted values and true values respectively versus time (data point), and the other one having fitted values and residuals respectively versus time (data point). The RMSE results with cross-validation for test data and trained data are reported in Form 2. The plotted results are shown in Figure 11-12.

Data Type	RMSE with cross-validation
Test	0.1036758
Train	0.1035854

Form 2 – RMSE Result with cross-validation (Linear Regression, Non-standardized)

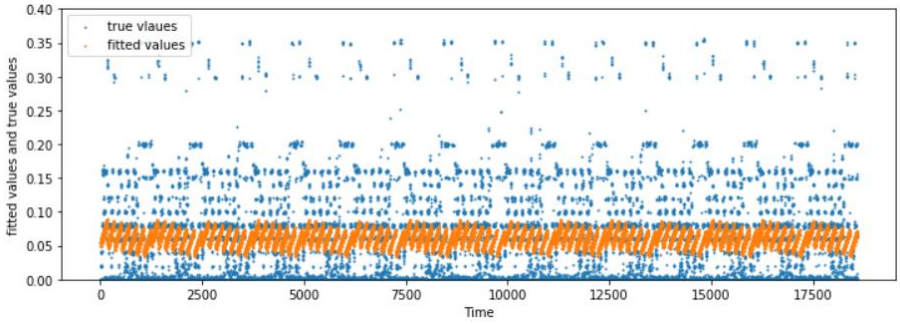

Figure 11 – Plotted result of true and fitted values versus time (Linear Regression, Non-standardized)

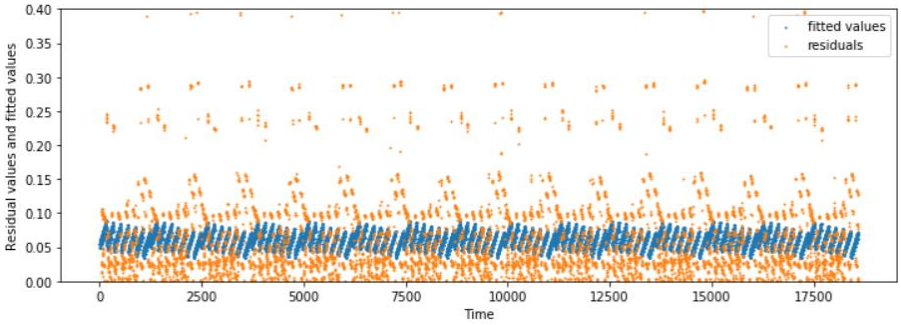

Figure 12 – Plotted result of residuals and fitted values versus time (Linear Regression, Non-standardized)

Then, to preprocess the data, we carried out standardization on the converted features, and repeated the fitting and evaluating procedure. The RMSE results with cross-validation for test data and trained data are reported in Form 3. The plotted results are shown in Figure 13-14.

Data Type	RMSE with cross-validation
Test	0.1036758
Train	0.1035854

Form 2 – RMSE Result with cross-validation (Linear Regression, Standardized)

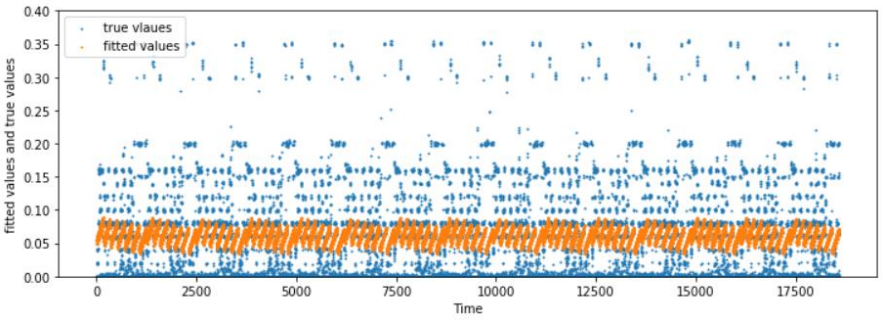

Figure 13 – Plotted result of true and fitted values versus time (Linear Regression, Standardized)

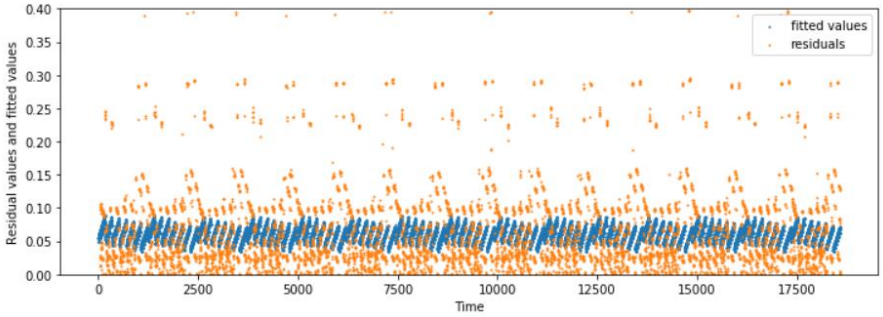

Figure 14 – Plotted result of residuals and fitted values versus time (Linear Regression, Standardized)

Comparing the results in question (i) and (ii), the RMSE results are the same, with the figures almost the same. We made an implication that the standardization has the attribute of linear transformation, resulting in equal proportional transformation for all data points. Therefore, the RMSE would be almost the same.

For feature selection, we used f_regression and mutual information regression measure to select three most important variables respectively. The function *feature_selection()* from sklearn can build a fitting model based on these two measures.

For f_regression measure, after training procedure, the results of reported scores, which evaluate the extent of importance, are shown in Form 3.

Features	Week #	Day of Week	Backup Start Time – Hour of Day	Work-Flow-ID	File Name
Result	8.45006257e-03	3.88163798e+01	1.50740934e+02	2.61386654e+01	2.53200943e+01

Form 3 – Reported Scores for 5 Features (f_regression)

From f_regression, three most important variables are “Backup Start Time – Hour of Day”, “Day of Week” and “Work-Flow-ID”. The next step is applying these three variables to train a new linear model, and repeat the RMSE reporting and plotting procedure in question (i). The reported RMSE and plotted results are separately shown in Form 4 and Figure 15, 16. Comparing to the RMSE in question (i), the RMSE in f_regression measure has a slight decrease in training data, indicating a better performance.

Data Type	RMSE with cross-validation
Test	0.1036707
Train	0.1035857

Form 4 – RMSE Result with cross-validation (Linear Regression, f_regression)

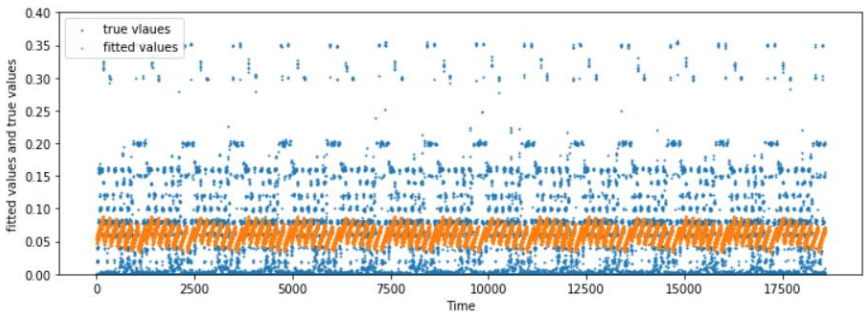

Figure 15 – Plotted result of true and fitted values versus time (f_regression)

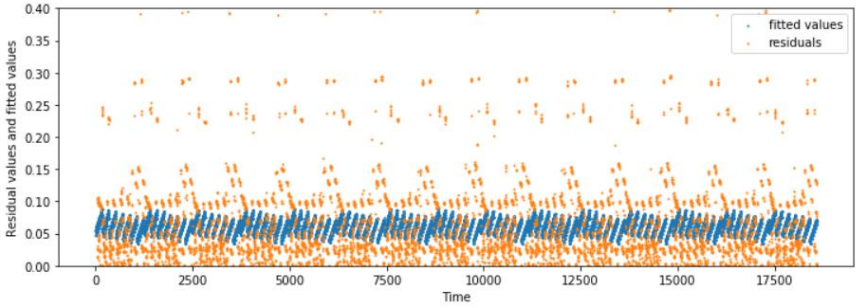

Figure 16 – Plotted result of residuals and fitted values versus time (f_regression)

With the same method, we can use mutual_info_regression to choose three most importance features. The results of reported scores are shown in Form 5.

Features	Week #	Day of Week	Backup Start Time – Hour of Day	Work-Flow-ID	File Name
Result	8.45006257e-03	3.88163798e+01	1.50740934e+02	2.61386654e+01	2.53200943e+01

Form 5 – RMSE Result with cross-validation (mutual_info_regression)

From mutual_info_regression, three most important variables are “Backup Start Time – Hour of Day”, “Work-Flow-ID” and “File Name”. When applying these three variables, the reported RMSE and plotted results are shown separately in Form 6 and Figure 17, 18. However, the RMSE results in training and testing data are higher than those in question (i), indicating a worse performance.

Data Type	RMSE with cross-validation
Test	0.1037723
Train	0.1036945

Form 4 – RMSE Result with cross-validation (Linear Regression, mutual_info_regression)

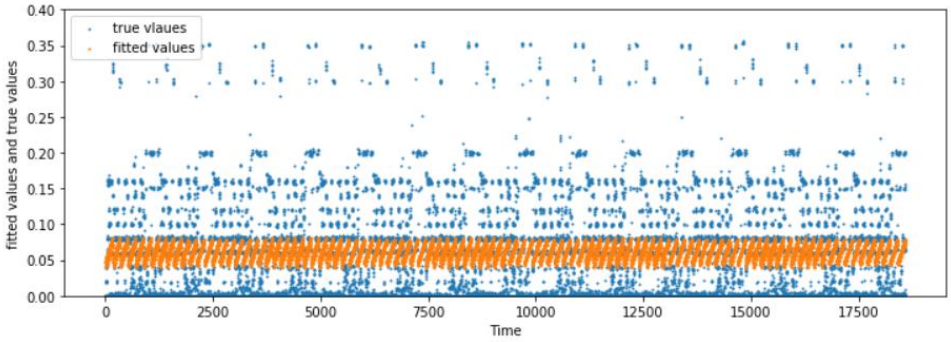

Figure 17 – Plotted result of true and fitted values versus time (mutual_info_regression)

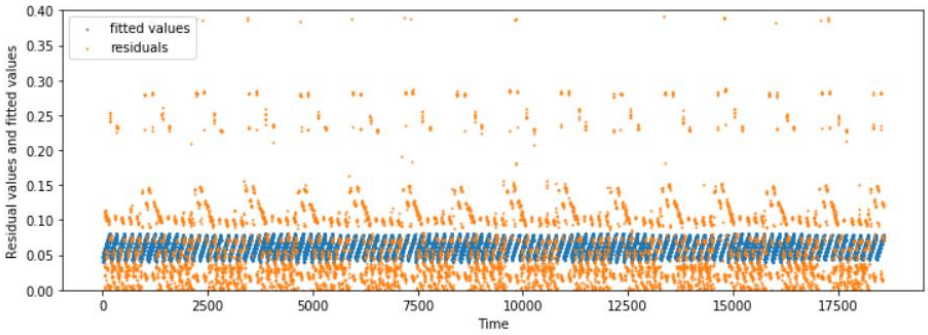

Figure 18 – Plotted result of residuals and fitted values versus time (mutual_info_regression)

In question (iv), we are supposed to evaluate different combinations of encoding schemes, and find the optimal combo. From the previous demonstration, the number of possible encoding combinations is 32. For this question, we evaluated all 32 possible combinations' training data RMSE and testing data RMSE, with RMSE plotted in range 1 to 32.

We converted the features of the data ("Week #", "Day of Week", "Backup Start Time – Hour of Day", "Work-Flow-ID", "File Name") into a set of one-digit numerical set in range 0 to 4, with step's value equal to 1. When evaluating the minimum RMSE, the numerical set of applying One-Hot-Encoder would be reported, and the unreported features are the subject to apply scalar encoding.

For training data RMSE, the plotted result for Train_RMSE versus combination is plotted in Figure 19. The detailed RMSE results for each combination are shown in Jupyter Notebook file.

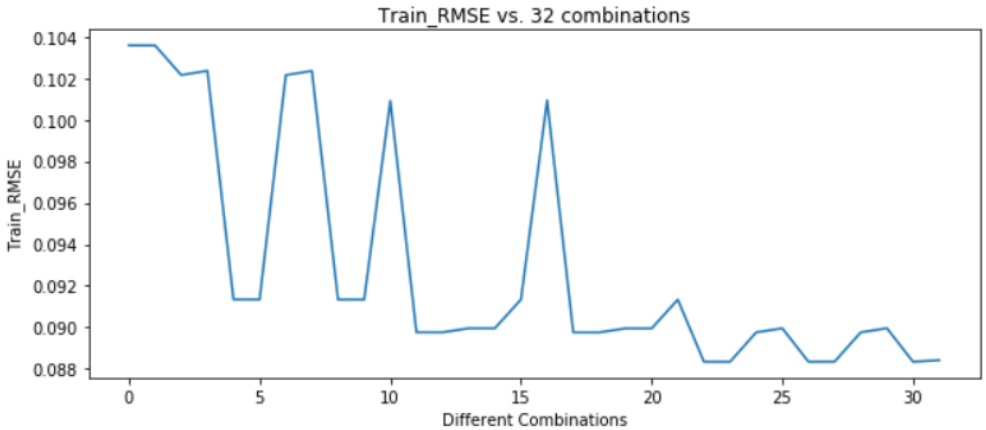

Figure 19 – Plotted result for Train_RMSE versus combination

According to the RMSE results, the minimum training data RMSE is 0.088338, and the corresponding best combo is (1, 2, 4), which means the best combination is using One-Hot-Encoder for feature "Day of Week", "Backup Start Time – Hour of Day", and "File Name", and using scalar encoding for feature "Week #" and "Work-Flow-ID".
For testing data RMSE, the plotted result for Test_RMSE versus combination is plotted in Figure 20. The detailed RMSE results for each combination are shown in Jupyter Notebook file.

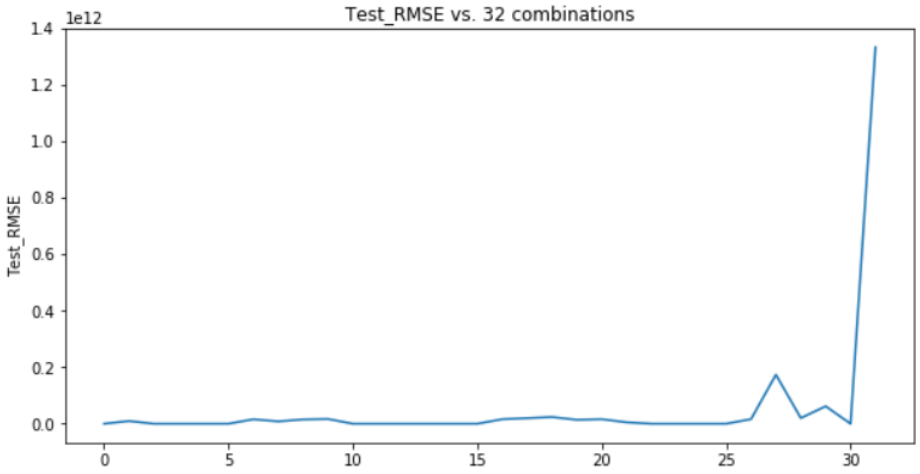

Figure 20 – Plotted result for Test_RMSE versus combination

According to the RMSE results, the minimum training data RMSE is 0.088506, and the corresponding best combo is (1, 2, 4), which means the best combination is also using One-Hot-Encoder for feature "Day of Week", "Backup Start Time – Hour of Day" and "File Name", and using scalar encoding for feature "Week #" and "Work-Flow-ID".
The testing data RMSE, training data RMSE, plotted results with fitted values/true values versus time, and residuals and fitted values versus time are separately shown in Form 5, Figure 21 and Figure 22.

Data Type	RMSE with cross-validation
Test	0.0885065
Train	0.0883380

Form 5 – RMSE Result with cross-validation (optimal encoding combination)

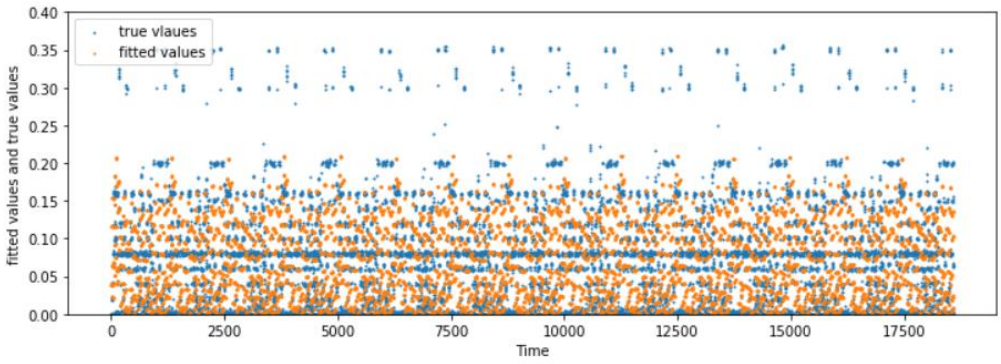

Figure 21 – Plotted result of true and fitted values versus time (optimal encoding combination)

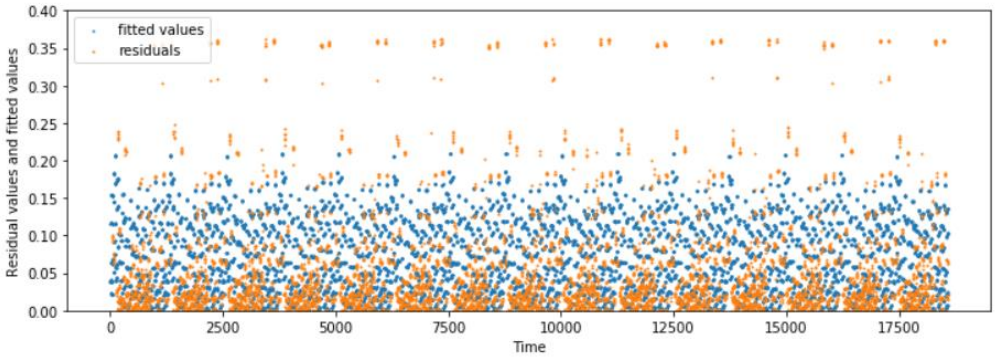

Figure 22 – Plotted result of residuals and fitted values versus time (optimal encoding combination)

From the RMSE results and figures, we can witness a significant improvement in performance. For the optimal performance set, we made an implication that the different attribute of One-Hot-Encoder and scalar encoding is the primary reason. One-Hot-Encoder would convert each feature into a binary vector, containing only one "1" element with other "0" elements, indicating an obvious bias and more converted features and a binary calculation based on a balanced matrix. Scalar encoding would convert each feature into consecutive integers within a certain range, making the calculation only proceeded in numerical way. When we apply One-Hot-Encoder to features with more relevance to the backup size, the performance tends to improve, since One-Hot-Encoder would enable the prediction to carry out with more features evaluated and estimated.

However, a significant increase in RMSE could be observed from several features in Figure 20. In the previous questions, we applied the ordinary least square as the penalty function.

In a linear regression fitting problem, we are supposed to find a vector **x** to qualify:

$$A\vec{x} = \vec{b}$$

For the case in this project, when there's no x or more than one x that qualifies this equation, the ordinary least square estimation will be over-fitted, amplifying some possible errors caused by biased original data and causing a severely biased or mistaken fitted data. Therefore, we might get a huge RMSE result.

To solve this problem, we applied following regularizations with several parameters swept for a relatively reasonable performance, which is evaluated by a minimum RMSE result:

1. Ridge Regularization: $\min_{\beta} \|Y - X\beta\|^2 + \alpha\|\beta\|_2^2$

2. Lasso Regularization: $\min_{\beta} \|Y - X\beta\|^2 + \alpha\|\beta\|_1$

3. Elastic Net Regularization: $\min_{\beta} \|Y - X\beta\|^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2$

<1> Ridge Regularization

The testing data RMSE result with different α value in Ridge Regularization is shown in Figure 23.

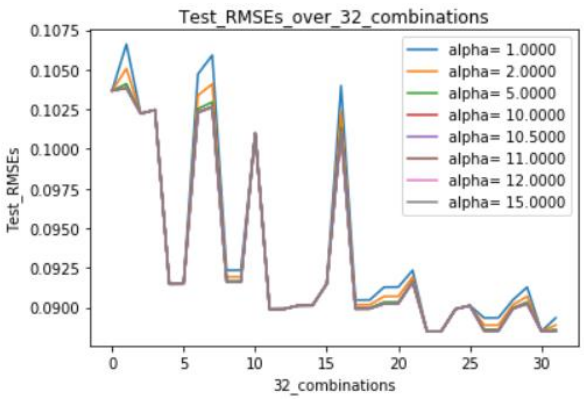

Figure 23 – Testing data RMSE with different α values (Ridge Regularization)

For Ridge Regularization, according to the sweeping result, when α equals to 12, we can obtain the optimal performance with One-Hot-Encoding applied to "Day of Week", "Backup Start Time: Hour of Day", "Work-Flow-ID", and scalar encoding applied to "Week #", "File Name". The reported RMSE, plotted results with fitted values/true values versus time, and residuals and fitted values versus time are separately shown in Form 6, Figure 24 and Figure 25.

Data Type	RMSE with cross-validation
Test	0.0885040
Train	0.0883377

Form 6 – RMSE Result with cross-validation (Ridge Regularization)

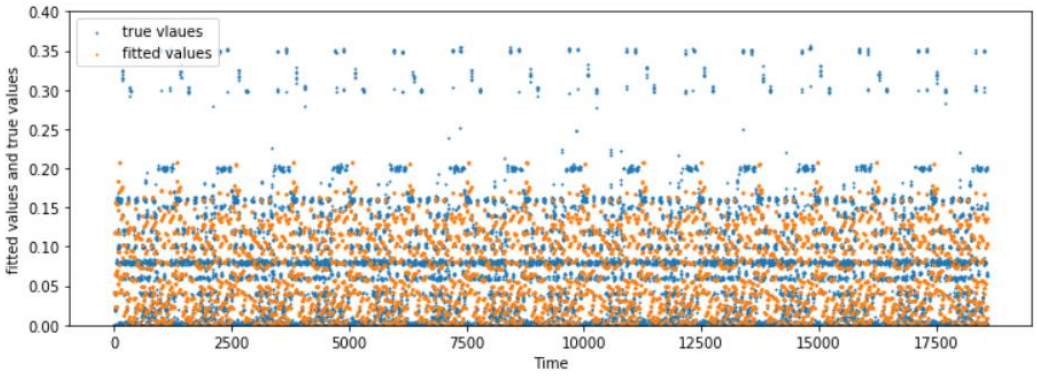

Figure 24 – Plotted result of true and fitted values versus time (Ridge Regularization)

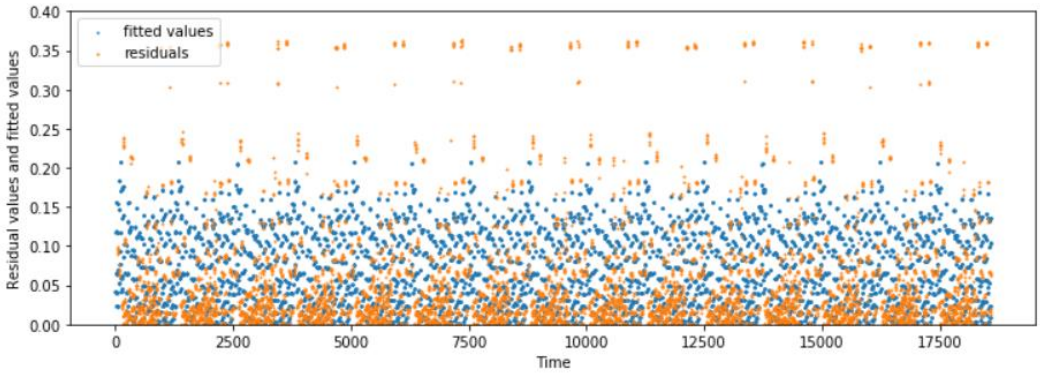

Figure 25 – Plotted result of residuals and fitted values versus time (Ridge Regularization)

<2> Lasso Regularization

The testing data RMSE result with different α value in Lasso Regularization is shown in Figure 26.

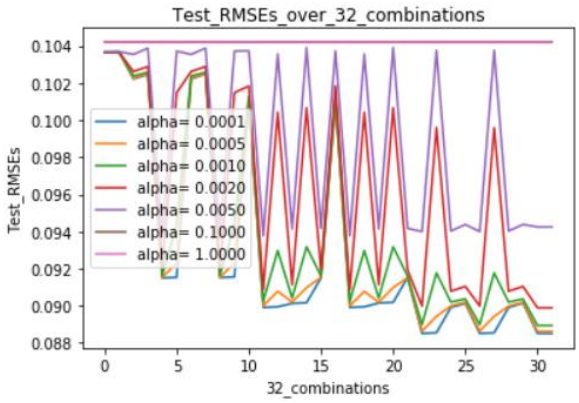

Figure 26 – Testing data RMSE with different α values (Lasso Regularization)

For Ridge Regularization, according to the sweeping result, when α equals to 0.0001, we can obtain the optimal performance with One-Hot-Encoding applied to all 5 features. The reported RMSE, plotted results with fitted values/true values versus time, and residuals and fitted values versus time are separately shown in Form 7, Figure 27 and Figure 28.

Data Type	RMSE with cross-validation
Test	0.0885082
Train	0.0883430

Form 7 – RMSE Result with cross-validation (Lasso Regularization)

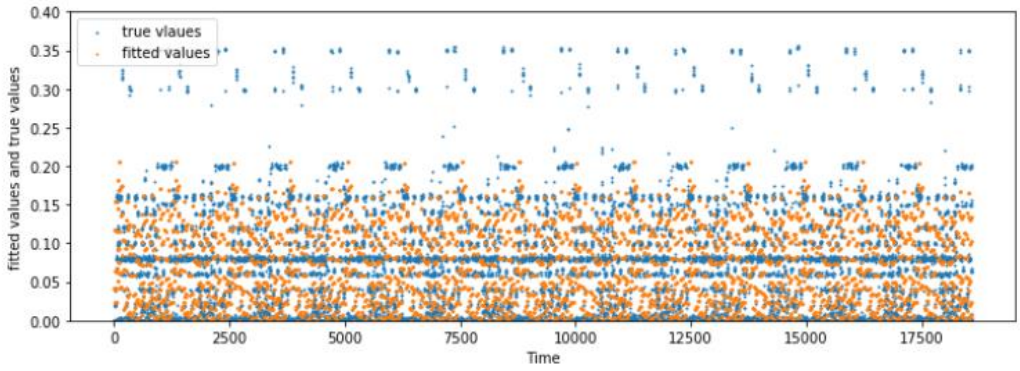

Figure 27 – Plotted result of true and fitted values versus time (Lasso Regularization)

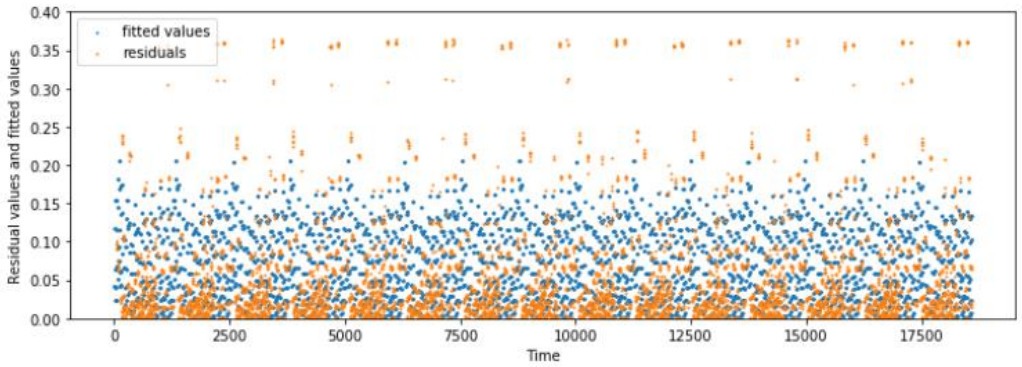

Figure 28 – Plotted result of residuals and fitted values versus time (Lasso Regularization)

<3> Elastic Net Regularization

For Elastic Net Regularization's mathematical expression:

$$\min_{\beta} \|Y - X\beta\|^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2$$

Here we assume:

$$\lambda_1 = \alpha c$$
$$\lambda_2 = \alpha(1 - c)$$

, where c represents L1 regression ratio. Due to the space limitation, the testing data RMSE result with different α and c value in Elastic Net Regularization is shown in Jupyter Notebook file.

According to the sweeping result, when α equals to 1, and c equals to 0.01, we can obtain the optimal performance with One-Hot-Encoding applied to "Work-Flow-ID", "File Name", and scalar encoding applied to "Week #", "Day of Week", "Backup Start Time – Hour of Day". The reported RMSE, plotted results with fitted values/true values versus time, and residuals and fitted values versus time are separately shown in Form 8, Figure 29 and Figure 30.

Data Type	RMSE with cross-validation
Test	0.1030802
Train	0.1030681

Form 8 – RMSE Result with cross-validation (Elastic Net Regularization)

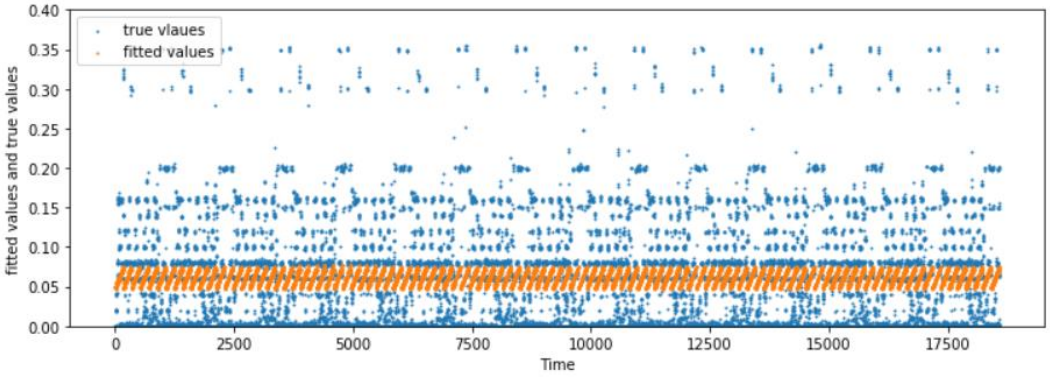

Figure 29 – Plotted result of true and fitted values versus time (Elastic Net Regularization)

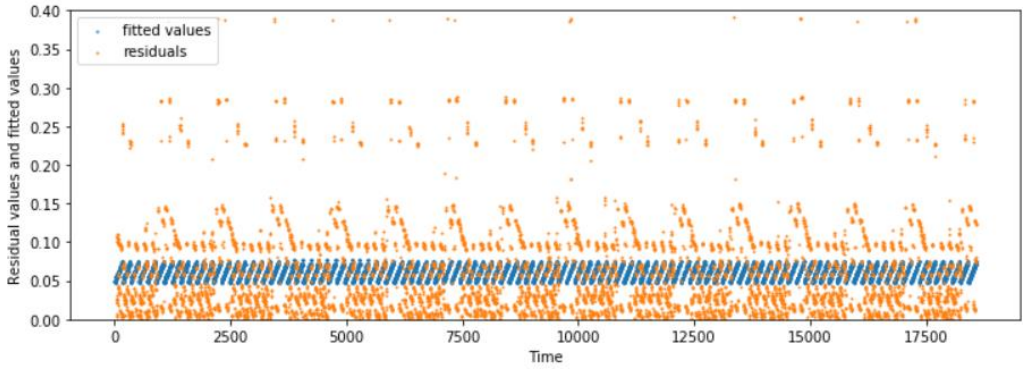

Figure 30 – Plotted result of residuals and fitted values versus time (Elastic Net Regularization)

The comparison of testing data RMSE between "optimal" combination in question (iv) and three regularization methods in question (v) is shown in Form 9. From the results shown in Form 9 and Figures, Ridge Regularization and Lasso Regularization have an approximative performance to that in ordinary mean square estimation, while Elastic Net Regularization displays a worse performance.

Scheme	"Optimal" Combination for ordinary mean square	Ridge Regularization	Lasso Regularization	Elastic Net Regularization
RMSE Result	0.0885065	0.0885040	0.0885082	0.1030802

Form 9 – RMSE Result Comparison between different schemes

Part (b) Random Forest Regression Model

The significant feature of random forest regression algorithm is a branch-structure-based decision-making system. During the training process, the nodes on the "tree" only enables one decision that is based on only one feature, which minimized a chosen measure of impurity. For a regression task, the impurity is evaluated by variance. When it comes to the importance of each feature, the value would be the averaged decreased variance for each node split with this feature in the forest and weighted by the number of samples it splits.

Another term that needs to be introduced in random forest algorithm is out of bag error. For each tree in random forest, only a part of the whole dataset is used to build the tree structure, and the rest of data can be used as the test set. Therefore, one can then define prediction-RMSE for each tree and calculate the average value over all trees. In the sklearn kit of Python, oob_score_ returns out of bag R^2 score, and (1-oob_score_) is the value of out of bag error.

The initial setting of the random forest regression model is shown in Form 10.

Parameter	Value
Number of trees	20
Depth of each tree	4
Bootstrap	True
Maximum number of features	5

Form 10 – Initial Setting of the Random Forest Regression Model

The first task in this section is reporting the training and average testing RMSE from 10-fold cross validation, and out of bag error from this model. The results are shown in Form 11 and Figure 31, 32.

Parameter	Value
Training RMSE	0.0605018500523
Testing RMSE	0.0606209313223
Out of Bag Error	0.661268519032

Form 11 – 10-fold cross validation results

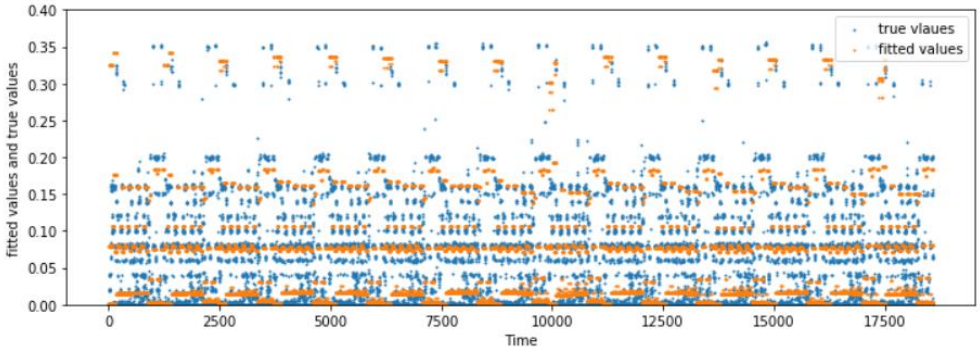

Figure 31 – Plotted result of true and fitted values versus time (Random Forest)

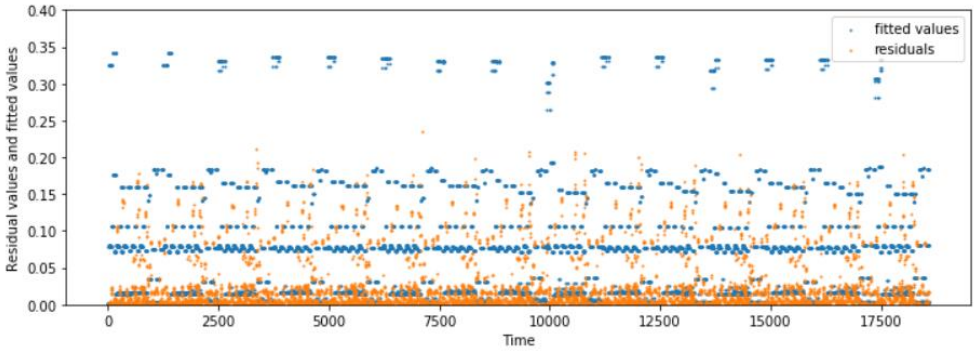

Figure 32 – Plotted result of residuals and fitted values versus time (Random Forest)

Next, we swept the number of trees from 1 to 200 and maximum number of features from 1 to 5. Under cross validation, the plotted results of bag error versus number of trees and average Test-RMSE versus number of trees are separately plotted in Figure 33 and Figure 34.

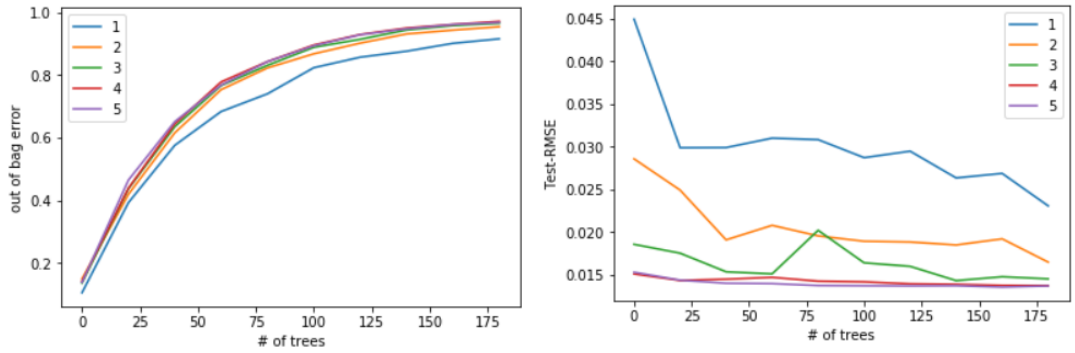

Figure 33 & 34 – Bag Error, average Test-RMSE versus Number of Trees

For further evaluation, we chose to evaluate the impact of maximum depth number to the performance. The maximum depth number was swept from 4 to 20, with a step value of 4. The plotted

results of bag error versus number of trees and average Test-RMSE versus number of trees with maximum depth number swept are separately plotted in Figure 35 and Figure 36.

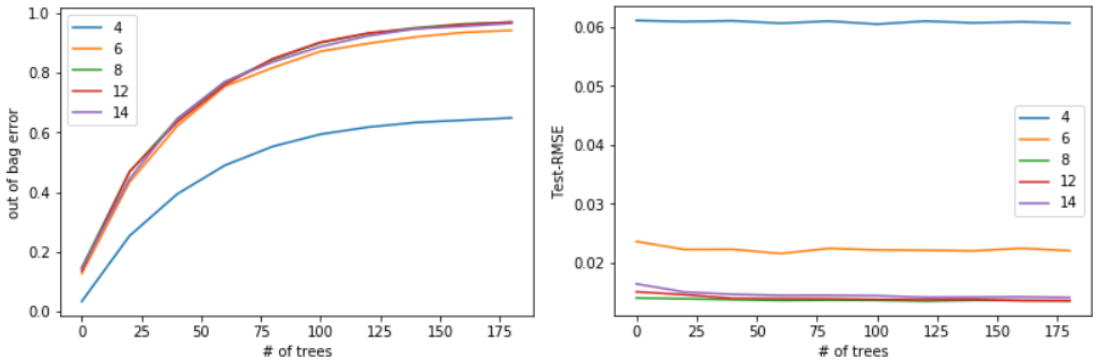

Figure 35 & 36 – Bag Error, average Test-RMSE versus Number of Trees (New maximum depth values)

From the graphs above, we can see that the number of features has a positive correlation with number of trees, indicating more features and trees have an improvement in performance. And when maximum depth number equals to 8, we can obtain the best solution. Therefore, we choose features = 5, number of trees = 200, and maximum depth number = 8 as the optimal parameters for the self-designed regression model.

The result of feature importances from this model is shown in Form 12.

Feature	Week #	Day of Week	Backup Start Time – Hour of Day	Work-Flow-ID	File Name
Value of Feature Importance	0.00352337	0.19761147	0.39543944	0.13035342	0.27307231

Form 12 – Value of Feature Importance Result (Self-designed Tree)

The final step of this section is the visualization of the random forest tree structure. When the maximum depth number is 4, the plotted result is shown in Figure 37, and the values of feature importance are shown in Form 13. The root node is located at the top of the whole tree structure.

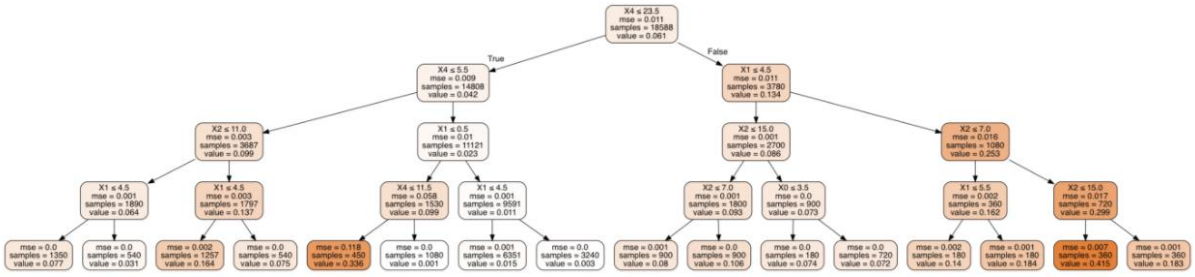

Figure 37 – Random Forest Tree Structure Layout

Feature	Week #	Day of Week	Backup Start Time – Hour of Day	Work-Flow-ID	File Name
Value of Feature Importance	6.40285059e-06	2.72288817e-01	1.49847701e-01	1.96205325e-01	3.81651754e-01

Form 13 – Value of Feature Importance Result

According to the tree structure graph above, the top 3 important features for maximum depth number's value equal to 4 is x4, x1, x2. The result got by attribute feature importance is x4, x1, x3. Therefore, we can draw a conclusion that the most important feature calculated by the decision tree is almost according to that reported by the regressor.

Part (c) Neural Network Regression Model

In this project, the neural network regression model is based on multi-layer perceptron, which is a supervised learning algorithm that learns a function:

$$f(\cdot): R^m \rightarrow R^o$$

, where m is the number of dimensions for input and o is the number of dimensions for output. With multi-layer perceptron, the performance of learning non-linear models or real-time (on-line learning) models would improve.

For the project purpose, we are supposed to use three activity functions: relu, logistic, and tanh. Another parameter is the number of hidden units. In this section, the number of hidden units was swept from 5 to 100, with the value of step equal to 5. All features were encoded via One-Hot-Encoder.

<1> ReLu

The plotted result for training RMSE and testing RMSE versus the number of hidden units with activity function ReLu is shown in Figure 38. (The blue curve refers to testing RMSE.)

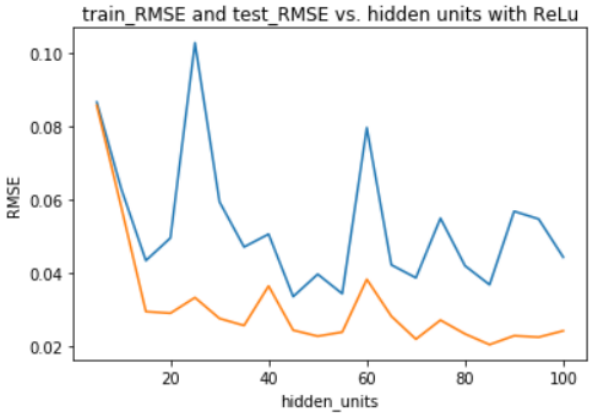

Figure 38 – Result of train_RMSE and test_RMSE versus number of hidden units (ReLu)

From Figure 38, when the number of hidden units is 45, the value of testing RMSE is minimized, indicating the optimal performance of the model under ReLu. The reported RMSE, plotted results with fitted values/true values versus time, and residuals and fitted values versus time are separately shown in Form 14, Figure 39 and Figure 40.

Data Type	RMSE with cross-validation
Test	0.0335226
Train	0.0243424

Form 14 – RMSE Result with cross-validation (Neural Network Model, ReLu)

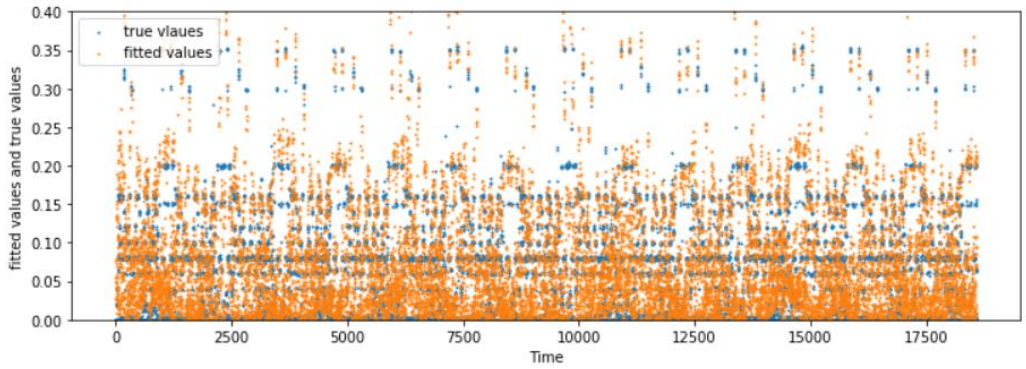

Figure 39 – Plotted result of true and fitted values versus time (Neural Network Model, ReLu)

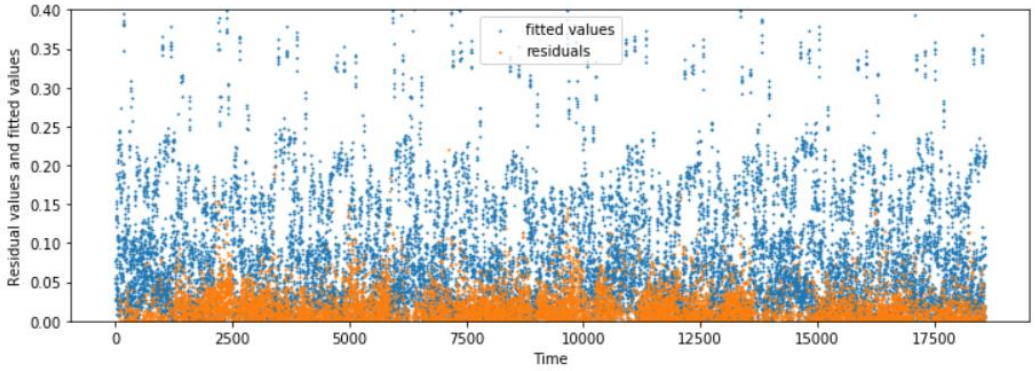

Figure 40 – Plotted result of residuals and fitted values versus time (Neural Network Model, ReLu)

<2> Logistic

The plotted result for training RMSE and testing RMSE versus the number of hidden units with activity function logistic is shown in Figure 41. (The blue curve refers to testing RMSE.)

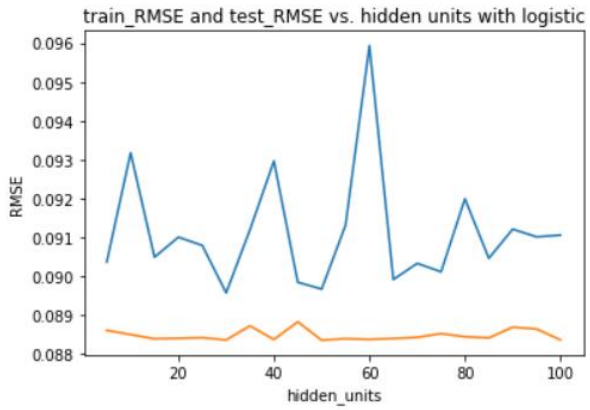

Figure 41 – Result of train_RMSE and test_RMSE versus number of hidden units (Logistic)

From Figure 41, when the number of hidden units is 30, the value of testing RMSE is minimized, indicating the optimal performance of the model under logistic. The reported RMSE, plotted results with fitted values/true values versus time, and residuals and fitted values versus time are separately shown in Form 15, Figure 42 and Figure 43.

Data Type	RMSE with cross-validation
Test	0.0895782
Train	0.0883659

Form 14 – RMSE Result with cross-validation (Neural Network Model, Logistic)

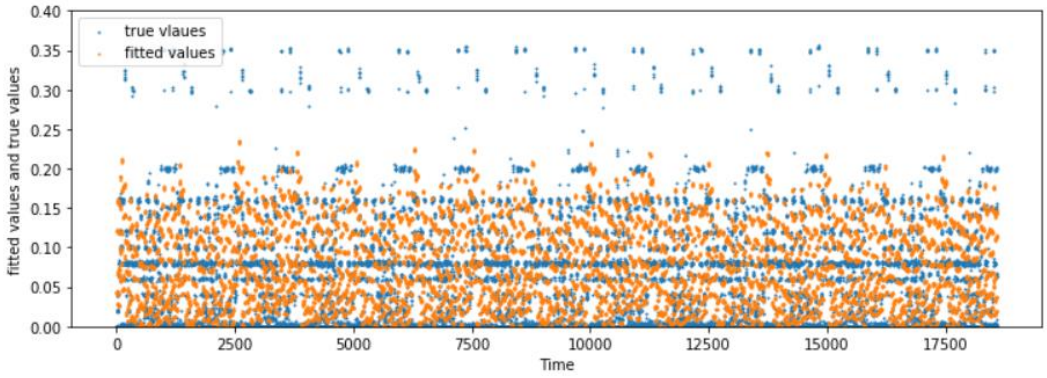

Figure 42 – Plotted result of true and fitted values versus time (Neural Network Model, Logistic)

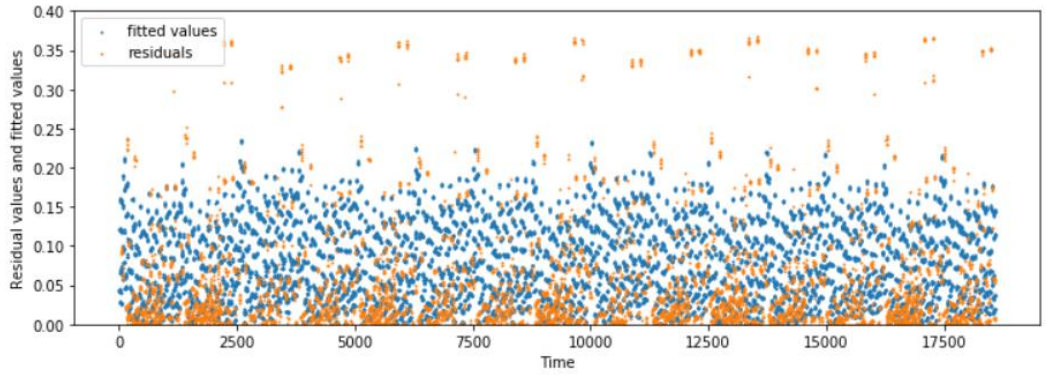

Figure 43 – Plotted result of residuals and fitted values versus time (Neural Network Model, Logistic)

<3> tanh

The plotted result for training RMSE and testing RMSE versus the number of hidden units with activity function tanh is shown in Figure 44. (The blue curve refers to testing RMSE.)

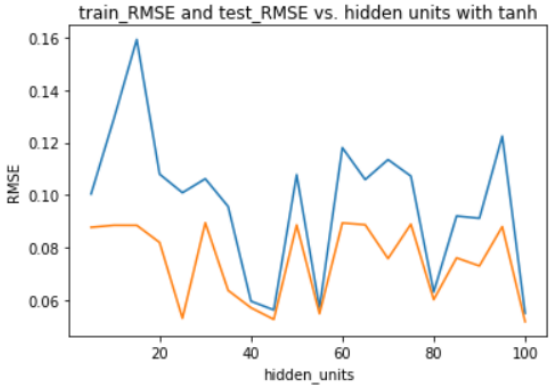

Figure 44 – Result of train_RMSE and test_RMSE versus number of hidden units (tanh)

From Figure 44, when the number of hidden units is 100, the value of testing RMSE is minimized,

indicating the optimal performance of the model under tanh. The reported RMSE, plotted results with fitted values/true values versus time, and residuals and fitted values versus time are separately shown in Form 16, Figure 45 and Figure 46.

Data Type	RMSE with cross-validation
Test	0.0895782
Train	0.0883659

Form 16 – RMSE Result with cross-validation (Neural Network Model, tanh)

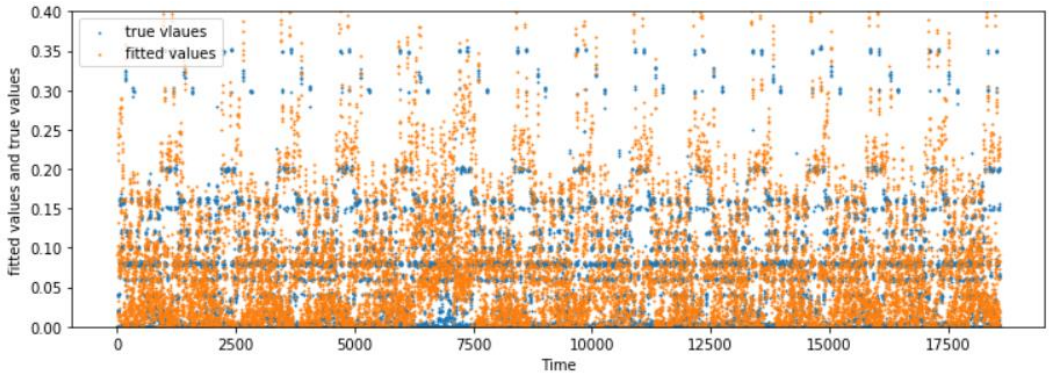

Figure 45 – Plotted result of true and fitted values versus time (Neural Network Model, tanh)

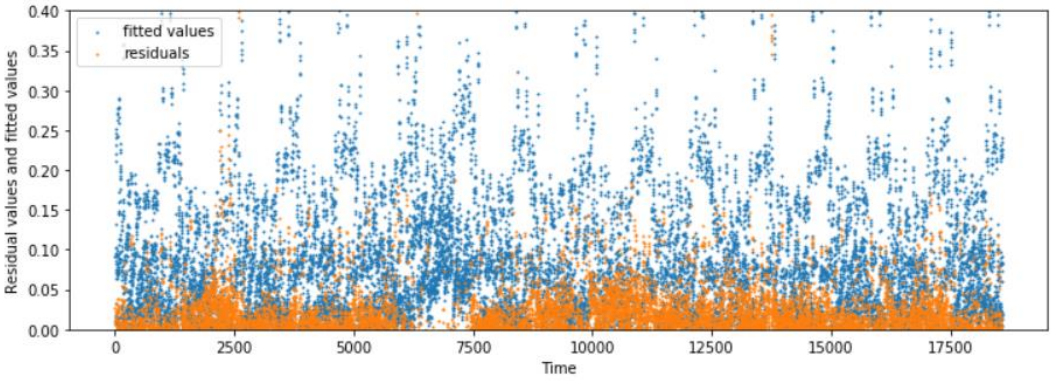

Figure 46 – Plotted result of residuals and fitted values versus time (Neural Network Model, tanh)

Part (d) Backup Size Prediction for each Workflow

In this section, we were supposed to make a prediction of backup size for each workflow with different schemes.

First, we applied the linear regression model. The RMSE result and the corresponding plots are separately shown in Form 17 and Figure 47-56.

Workflow ID	Test_RMSE	Train_RMSE
Work_flow_0	0.0358870	0.0358355
Work_flow_1	0.1489186	0.1487660
Work_flow_2	0.0430669	0.0429093
Work_flow_3	0.0072609	0.0072439
Work_flow_4	0.0859906	0.0859219

Form 17 - RMSE Result of all 5 workflows

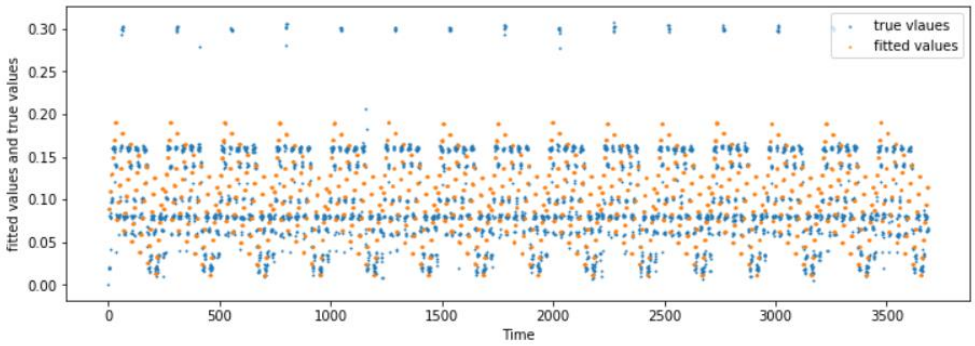

Figure 47 – Plotted result of true and fitted values versus time (Linear Regression, workflow 0)

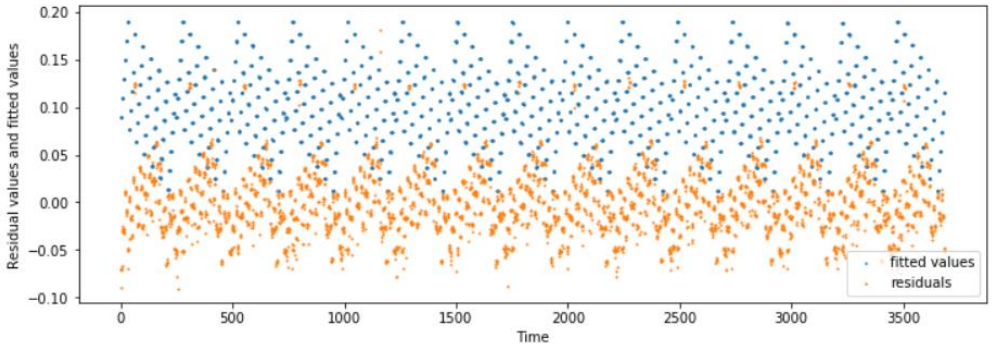

Figure 48 – Plotted result of residuals and fitted values versus time (Linear Regression, workflow 0)

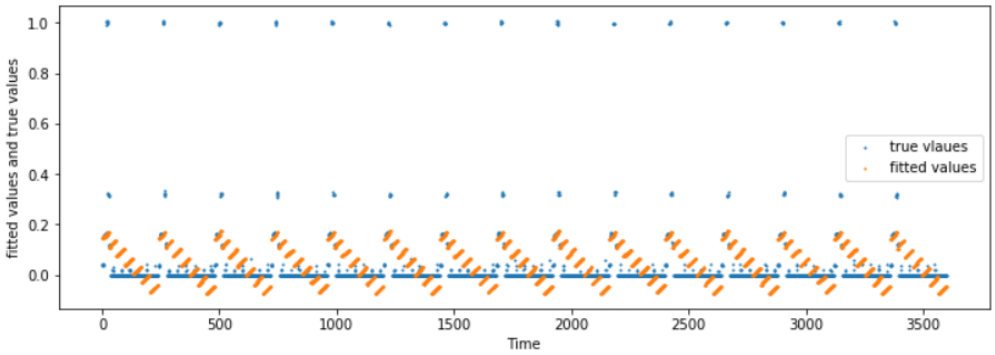

Figure 49 – Plotted result of true and fitted values versus time (Linear Regression, workflow 1)

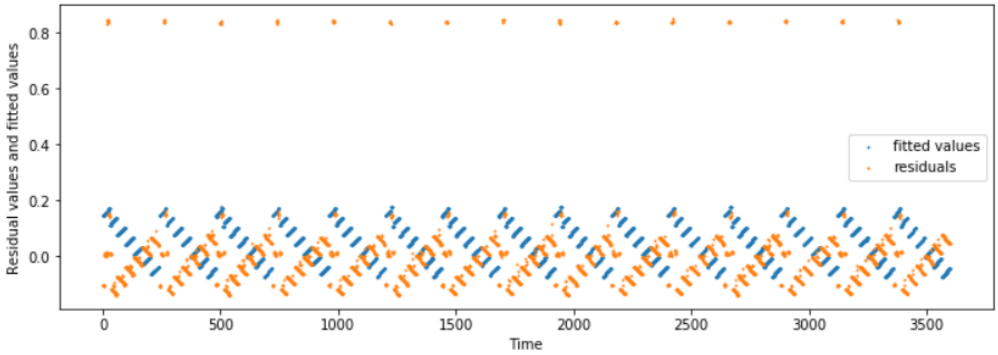

Figure 50 – Plotted result of residuals and fitted values versus time (Linear Regression, workflow 1)

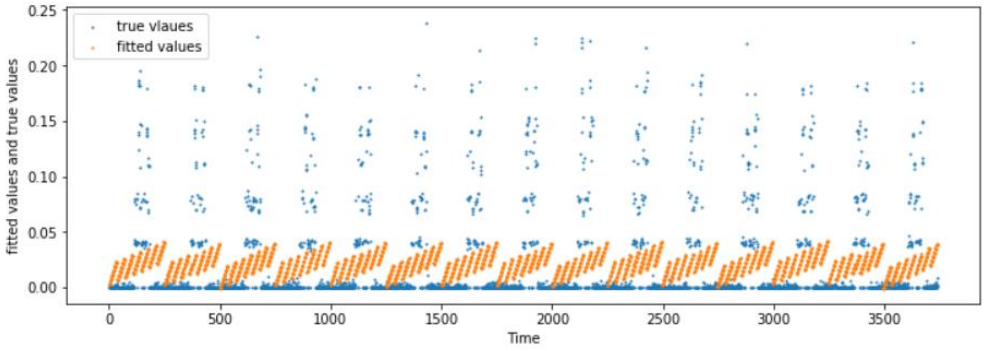

Figure 51 – Plotted result of true and fitted values versus time (Linear Regression, workflow 2)

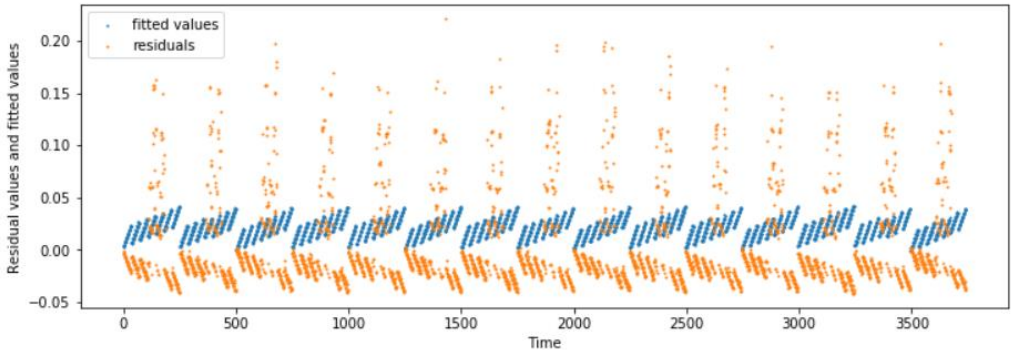

Figure 52 – Plotted result of residuals and fitted values versus time (Linear Regression, workflow 2)

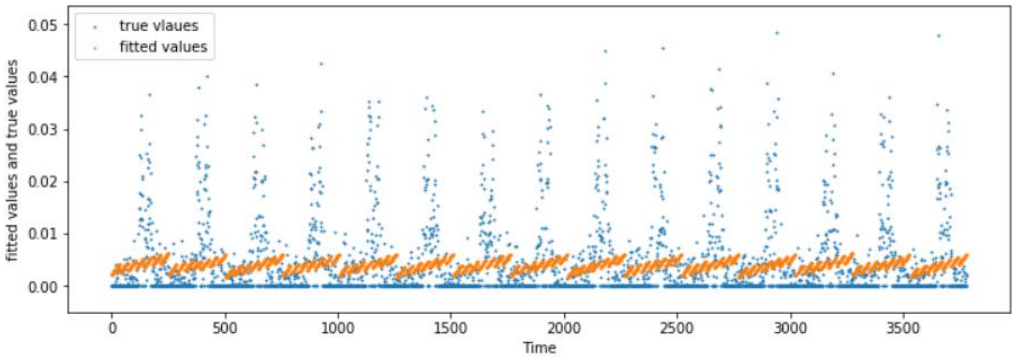

Figure 53 – Plotted result of true and fitted values versus time (Linear Regression, workflow 3)

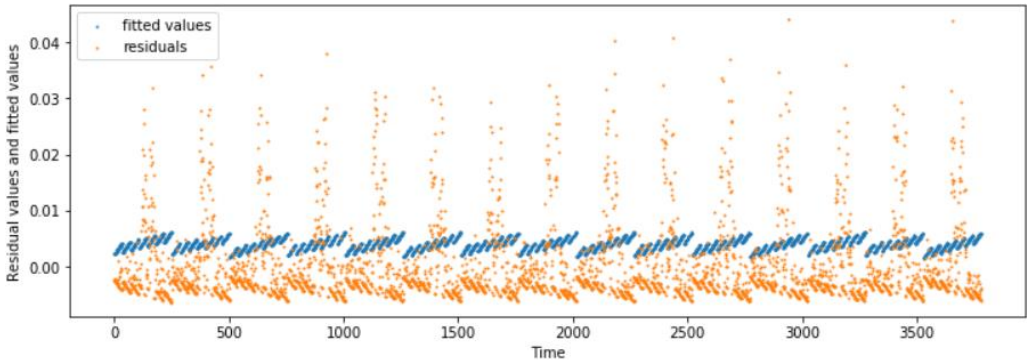

Figure 54 – Plotted result of residuals and fitted values versus time (Linear Regression, workflow 3)

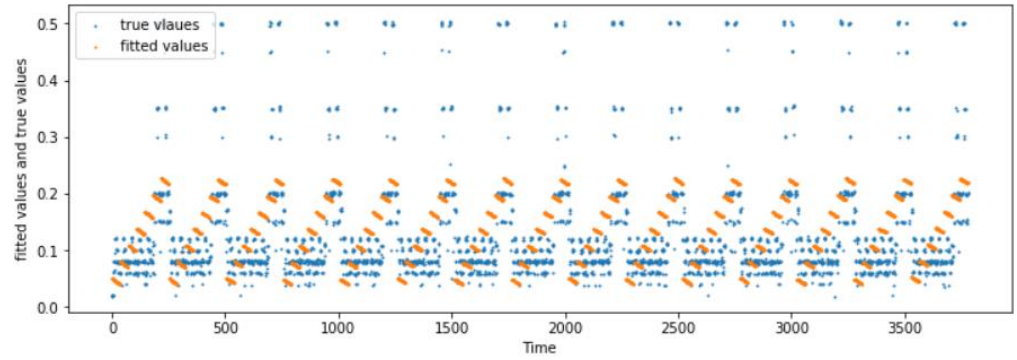

Figure 55 – Plotted result of true and fitted values versus time (Linear Regression, workflow 4)

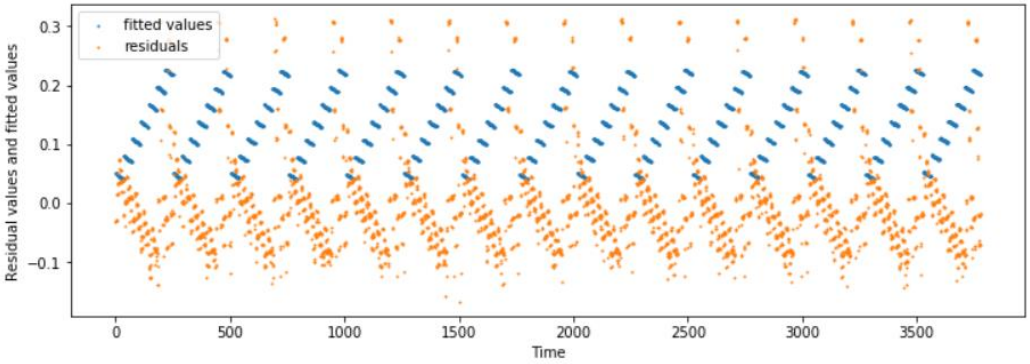

Figure 56 – Plotted result of residuals and fitted values versus time (Linear Regression, workflow 4)

From the RMSE results and graphs shown above, comparing the results of single workflow with that of the whole dataset, the change in performance is uneven. A decrease in performance could be witnessed in workflow 1, while the performance in other 4 workflows improves slightly.

Linear regression model tends to predict the backup size with a consistent pattern in data change. When we made the prediction to the whole dataset, since the dataset consists all workflows with no regular patterns, a prediction based on a certain pattern would be less reliable, causing more possibility to generate huge prediction error. Nonetheless, the prediction within each workflow sets a category limitation, allowing the prediction to execute only based on the data in one workflow. Therefore, a smaller prediction error would be witnessed when predicting with a single workflow.

Next, we used polynomial function to carry out a more complex regression model on the dataset. In this section, the parameter we adjusted is the degree of polynomial. We still used 10-fold cross validation as the method of evaluation. The plotted result of the average train and test RMSE of the trained model versus the degree of polynomial is shown in Figure 57.

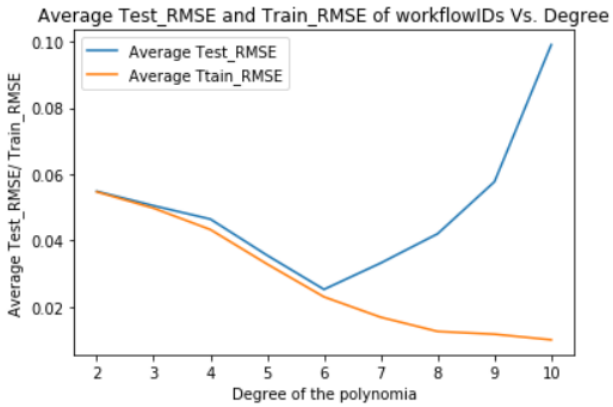

Figure 57 - Result of the average train and test RMSE versus degree of the polynomial

From the graph, we can identify a threshold from the test RMSE curve. When the degree of polynomial is equal to 6, the test RMSE is minimized, and when the degree goes higher, the value of test RMSE also gets higher, indicating a worse performance. Therefore, the threshold is 6. Under this threshold value, the RMSE result and the corresponding performance plots are separately shown in Form 18 and Figure 58-67.

Workflow ID	Test_RMSE	Train_RMSE
Work_flow_0	0.0128652	0.0103765
Work_flow_1	0.0430033	0.0424840
Work_flow_2	0.0273099	0.0218344
Work_flow_3	0.0053473	0.0045945
Work_flow_4	0.0375325	0.0354736

Form 18 - RMSE Result of all 5 workflows (with polynomial function)

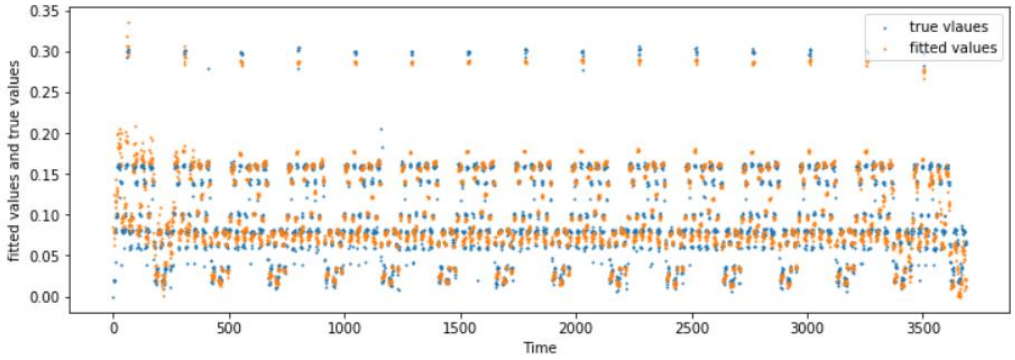

Figure 58 – Plotted result of true and fitted values versus time (Polynomial Regression, workflow 0)

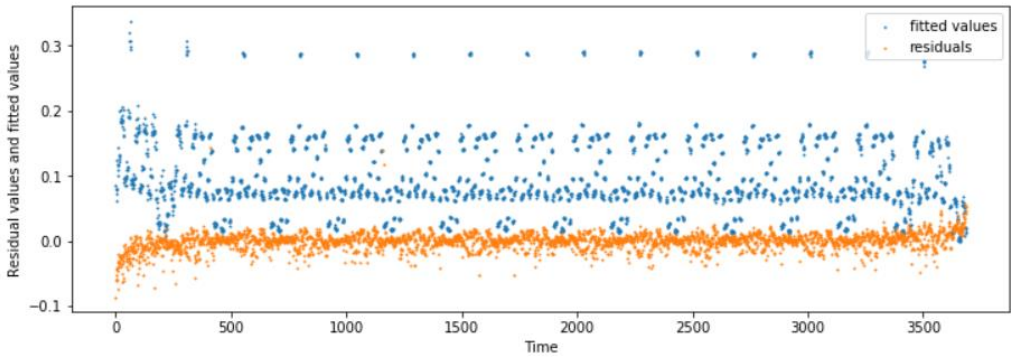

Figure 59 – Plotted result of residuals and fitted values versus time (Polynomial Regression, workflow 0)

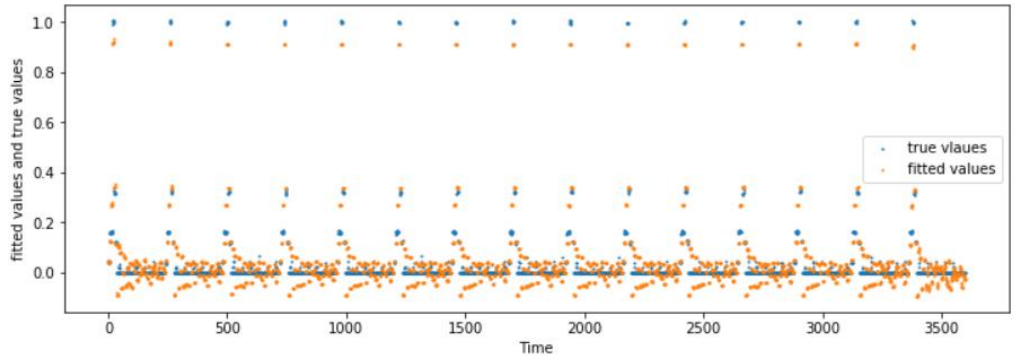

Figure 60 – Plotted result of true and fitted values versus time (Polynomial Regression, workflow 1)

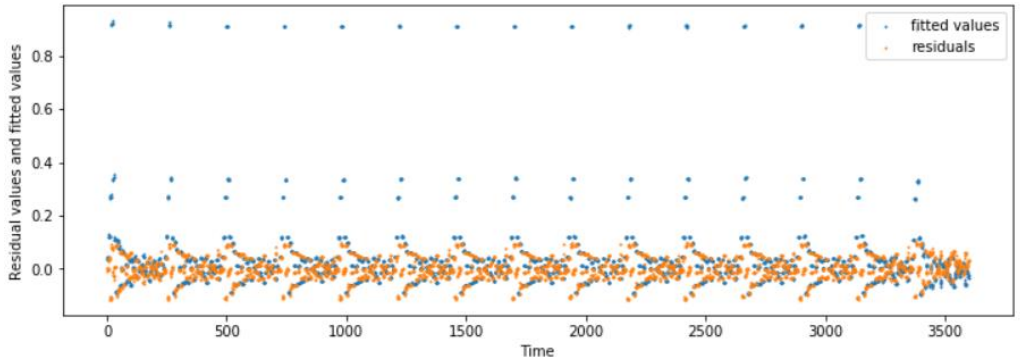

Figure 61 – Plotted result of residuals and fitted values versus time (Polynomial Regression, workflow 1)

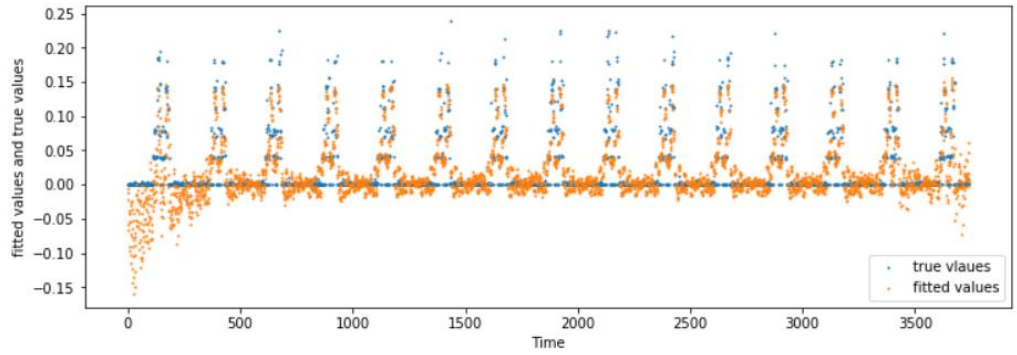

Figure 62 – Plotted result of true and fitted values versus time (Polynomial Regression, workflow 2)

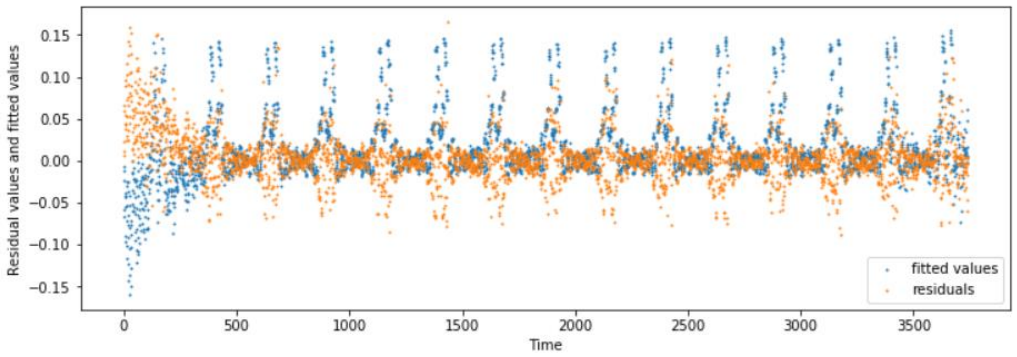

Figure 63 – Plotted result of residuals and fitted values versus time (Polynomial Regression, workflow 2)

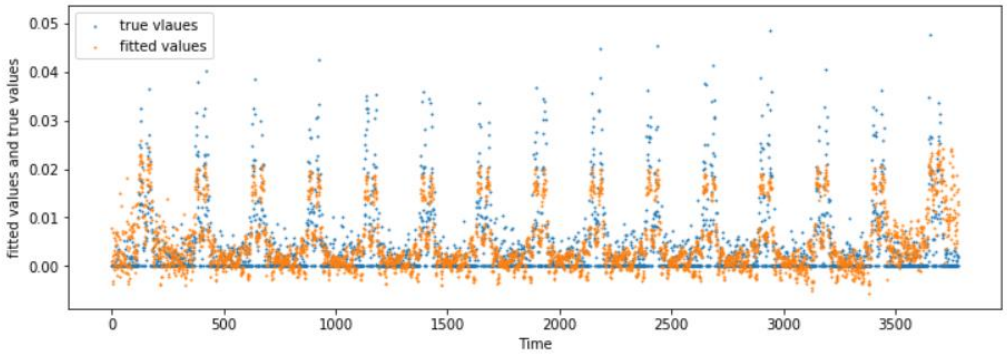

Figure 64 – Plotted result of true and fitted values versus time (Polynomial Regression, workflow 3)

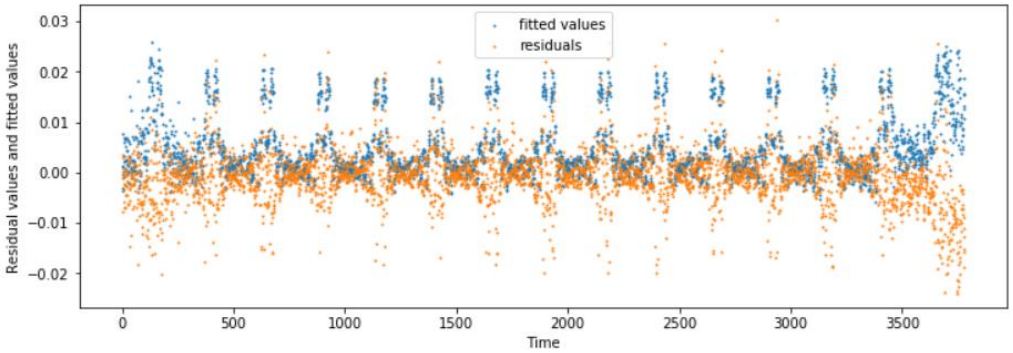

Figure 65 – Plotted result of residuals and fitted values versus time (Polynomial Regression, workflow 3)

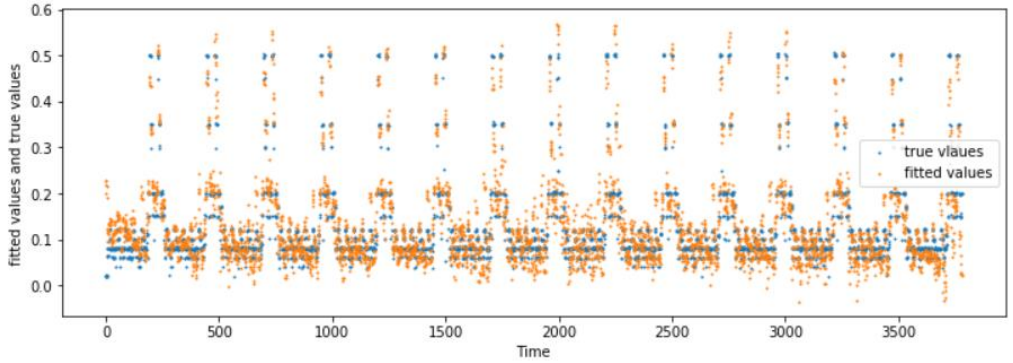

Figure 66 – Plotted result of true and fitted values versus time (Polynomial Regression, workflow 4)

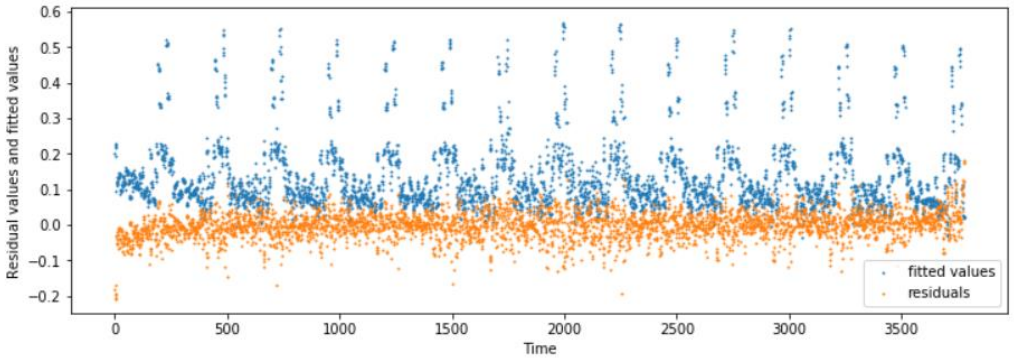

Figure 67 – Plotted result of residuals and fitted values versus time (Polynomial Regression, workflow 4)

From the figures above, we can witness a significant improvement in the performance when applying polynomial function.

Cross validation help controlling the complexity of the model in this section. For a model, the complexity refers to how hard the model is to learn from limited data. In a typical cross validation process, part of the training data would be used for the model performance estimation, with each procedure in the fitting process performed in each fold of the cross validation. In this case, the model would be less sensitive to extreme circumstances, such as over-fitting or ill-conditioning. In addition, the error calculated by the cross validation is a comprehensive result by the multiple folds or with multiple methods applied, providing an unbiased estimation with minimized cross validation error of the model performance evaluation.

Part (e) k-nearest Neighbor Regression

In this section, we used the regression method based on k-nearest neighbor regression. We swept the parameter k from 1 to 5 with a step value of 5.

Under different k values, the RMSE result and the corresponding performance plots are separately shown in Form 19, with the plotted result for test RMSE and train RMSE versus k shown in Figure 68.

K value	Test_RMSE	Train_RMSE
1	0.0201658	0.0000000
2	0.0343122	0.0289562
3	0.0363105	0.0299732
4	0.0374536	0.0284126
5	0.0433936	0.0269626

Form 18 - RMSE Result with different k values (k-NN regression)

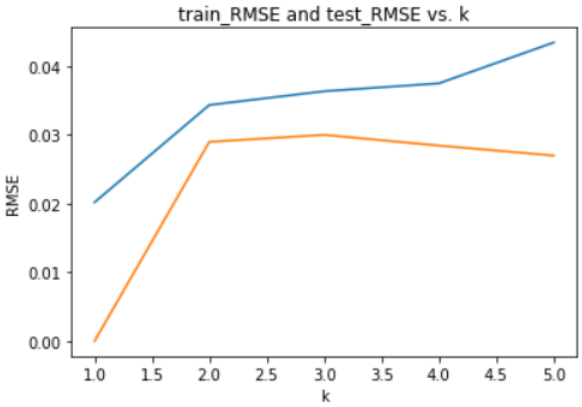

Figure 68 - Plotted result for test RMSE and train RMSE versus k (k-NN Regression)

From the graph and form above, when k=1, we can obtain an optimal performance of the model. The corresponding plotted results with fitted values/true values versus time, and residuals and fitted values versus time are separately shown in Figure 69 and Figure 70.

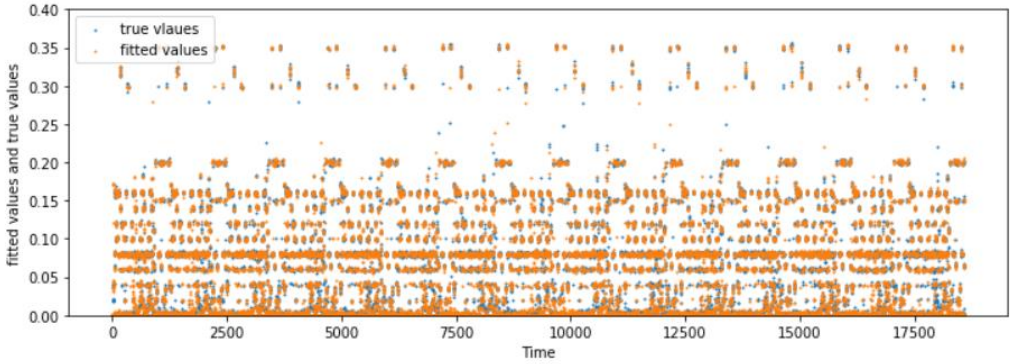

Figure 69 – Plotted result of true and fitted values versus time (k-NN Regression)

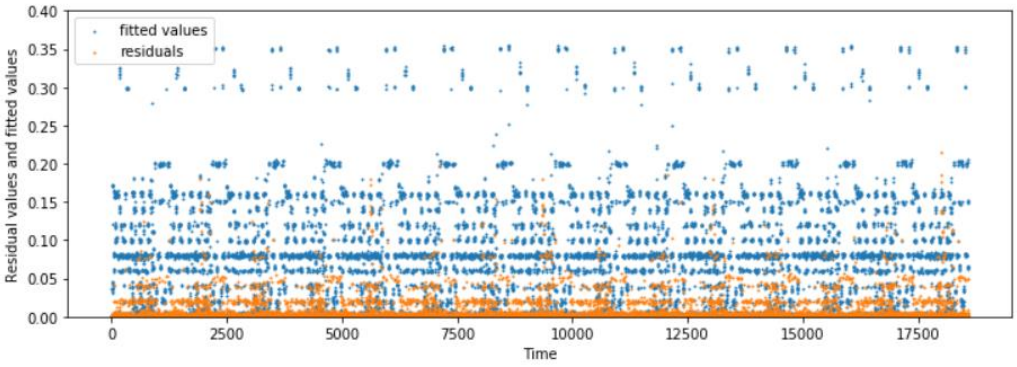

Figure 70 – Plotted result of residuals and fitted values versus time (k-NN Regression)

Performance Comparison

From all regression methods mentioned above, we can observe some differences between the performance of models based on these methods.

When coping with data with several categories, from the results, random forest regression shows a relatively optimal performance, since this method could use decision tree structure for the feature importance's ranking, giving an intuitive hint for the model design and estimation.

When the overall dataset is sparse, random forest regression, polynomial-based regression and k-NN regression (k=1) can provide a relatively better result.

For the overall performance, the random forest regression has an optimal performance.